Application No.: 09/730,489 **Atty Docket: JGR/CMRC 1003-1**

REMARKS

Claims 1-83 are pending in this application. The Examiner rejected all claims under 35 U.S.C. § 101 as directed to non-statutory subject matter. Examiner also rejected all claims under 35 U.S.C. § 102(e) as anticipated by Bowman Amuah, U.S. Patent No. 6,697,824. The rejections are traversed below.

Rejections Under 35 U.S.C. § 101

The Examiner has identified and acknowledged sufficient statutory subject matter in the claims to overcome any Section 101 issue. "[T]he steps of 'initiating a session... ., responding during the session..., a schema identifier for the additional message ..., and polymorphic response...'" are functional actions in a data exchange method or protocol. Once the Examiner has acknowledged that these actions involve conveying data between computers, Section 101 is fully satisfied, because these are limitations that are in the technological art.

Moreover, use of a polymorphic response message is functional, enhancing the flexibility of e-commerce sessions. The controlling principles for judging functionality are articulated in *In re Lowrey*, 32 F.3d 1579, 1581 (Fed. Cir. 1994), which is cited in MPEP § 2106 at 2100-12 (Rev. 2, May 2004). *In re Lowry* presented a claimed data structure that was resident in memory, including information resident in a database that was accessible to and used by an application program. The claimed data structure claimed defined how other data objects interrelated. The claim did not include a particular application program or any method steps. The data structure was functional because it defined data relationships among other data objects.

The *In re Lowry* court reasoned on page 1583:

> "Lowry's ADOs do not represent merely underlying data in a database. ADOs contain both information used by application programs and information regarding their physical interrelationships within a memory. Lowry's claims dictate how application programs manage information. Thus, Lowry's claims define functional characteristics of the memory.
>
> ... the claims require specific electronic structural elements which impart a physical organization on the information stored in memory. Lowry's invention manages information. As Lowry notes, the data structures provide increased computing efficiency."

From this passage, the rule emerges that a data structure accessible in memory (see, Section 2106 of the MPEP, page 2100-12) is patentable subject matter when it contains both information used by application programs and information regarding physical interrelationships within memory. That is, a data structure accessible in memory is functional if it dictates how application programs manage information.

Claim 1 includes,

> *responding during the session with an additional message including:*
>
> > *a schema identifier for the additional message, resolvable in a context of a system identifier; and*
> >
> > *a polymorphic response ~~comprising~~ including a type and [[a]] version, ~~wherein the polymorphic response including~~ and one or more additional data elements corresponding to values assigned to the type and version.*

The schema identifier resolvable in a context of a system identifier, the type and version of the claimed method "impart a physical organization on the information". The method manages information. The schema, type and version increase computing efficiency and flexibility.

Both because the Examiner has acknowledged substantial functional actions in this method claim and because, under *In re Lowry* and MPEP § 2016, the schema, type and version are functional, Applicants respectfully submit that the rejection under Section 101 should be withdrawn.

Should the Examiner have further issues about application of Section 101, Applicants respectfully request a three-way interview including Quality Control Examiner Jack Harvey, (571) 272-3896. Examiner Harvey is a particularly knowledgeable Section 101 resource person.

Rejections Under 35 U.S.C. § 102(e)

The Examiner rejects claims 1-83 under 35 U.S.C. § 102(e) as being unpatentable over Bowman-Amuah, U.S. Patent No. 6,697,824.

We begin with **independent claims 1, 29, 47 and 64**, all of which use of a polymorphic initiation or response message, as set out above. Applicants review below the passages of Bowman-Amuah that mention polymorphism, whether or not the

Examiner cited them. The polymorphism in Bowman-Amuah does not meet the limitations of the claims.

Bowman-Amuah uses the word "polymorphism" four times, one of which the Examiner cited. The first discussion is in column 6, the word appearing in lines 19 & 23:

> When the object or class representing the ceramic piston engine inherits all of the aspects of the objects representing the piston engine, it inherits the thermal characteristics of a standard piston defined in the piston engine class. However, the ceramic piston engine object overrides these ceramic specific thermal characteristics, which are typically different from those associated with a metal piston. It skips over the original and uses new functions related to ceramic pistons. Different kinds of piston engines have different characteristics, but may have the same underlying functions associated with it (e.g., how many pistons in the engine, ignition sequences, lubrication, etc.). To access each of these functions in any piston engine object, a programmer would call the same functions with the same names, but each type of piston engine may have different/overriding implementations of functions behind the same name. This ability to hide different implementations of a function behind the same name is called polymorphism and it greatly simplifies communication among objects.
>
> With the concepts of composition-relationship, encapsulation, inheritance and polymorphism, an object can represent just about anything in the real world. In fact, logical perception of the reality is the only limit on determining the kinds of things that can become objects in object-oriented software. Some typical categories are as follows:
>
> Objects can represent physical objects, such as automobiles in a traffic-flow simulation, electrical components in a circuit-design program, countries in an economics model, or aircraft in an air-traffic-control system.
>
> Objects can represent elements of the computer-user environment such as windows, menus or graphics objects.
>
> An object can represent an inventory, such as a personnel file or a table of the latitudes and longitudes of cities.
>
> An object can represent user-defined data types such as time, angles, and complex numbers, or points on the plane.

One use refers to polymorphic function calls, within a programming module (line 19) and the other refers to polymorphic objects in a programming language (line 23). Nothing in the passage suggests sending polymorphic messages with the claimed structure between buyer and supplier applications in e-commerce.

The second discussion of polymorphism, in the context of C++ OOP programming principles, spans columns 6-7.

well as an increased speed of its development.

Programming languages are beginning to fully support the OOP principles, such as encapsulation, inheritance, polymorphism, and composition-relationship. With the advent of the C++ language, many commercial software developers have embraced OOP. C++ is an OOP language that offers a fast, machine-executable code. Furthermore, C++ is suitable for both commercial-application and systems-programming projects. For now, C++ appears to be the most popular choice among many OOP programmers, but there is a host of other OOP languages, such as Smalltalk, Common Lisp Object System (CLOS), and Eiffel. Additionally, OOP capabilities are being added to more traditional popular computer programming languages such as Pascal.

The benefits of object classes can be summarized, as follows:

Objects and their corresponding classes break down complex programming problems into many smaller, simpler problems.

Encapsulation enforces data abstraction through the organization of data into small, independent objects that can communicate with each other. Encapsulation protects the data in an object from accidental damage, but allows other objects to interact with that data by calling the object's member functions and structures.

Subclassing and inheritance make it possible to extend and modify objects through deriving new kinds of objects from the standard classes available in the system. Thus, new capabilities are created without having to start from scratch.

Polymorphism and multiple inheritance make it possible for different programmers to mix and match characteristics of many different classes and create specialized objects that can still work with related objects in predictable ways.

Class hierarchies and containment hierarchies provide a flexible mechanism for modeling real-world objects and the relationships among them.

At line 1, polymorphism is mentioned as one of the properties of OOP programming languages, such as C++, Smalltalk, CLOS and Eiffel. At line 29, polymorphism allows different programmers to mix and match characteristics of different classes to create specialized program objects, within a programming environment. These references to polymorphism do not teach or suggest sending polymorphic messages with the claimed

structure between buyer and supplier applications in e-commerce.

Again from claim 1 as amended, the limitation that polymorphism in Bowman-Amuah must meet is:

> *responding during the session with an additional message including:*
>
> > *a schema identifier for the additional message, resolvable in a context of a system identifier; and*
> >
> > *a polymorphic response including a type and version and one or more additional data elements corresponding to values assigned to the type and version.*

Applicants are not claiming the concept of polymorphism as was being applied by the C++ compliers described by Bowman-Amuah. Instead, Applicants have claimed a particular implementation of polymorphism for e-commerce messages. As explained in the specification, p. 24, "Use of a polymorphic data schema decouples programming from the evolution of document types and applications. Use of namespaces supports coordination of document type names, development, deployment and maintenance of applications."

Therefore, Applicants respectfully submit that the independent claims and the claims that depend from them are allowable over Bowman-Amuah.

Dependent **claims 2, 30, 48 and 65** require a system identifier explicit in the initiation or response message. The Examiner argues that there is an explicit system identifier in Payment reference 2604 and claims 1, 3. Reference 2604 is used in columns 60 and 62-64, set forth below.

Order Processing **1010**

FIG. **26** illustrates the Order Processing portion **1010** of the eCommerce Application Framework **1000**. Subsections include Merchandise Selection **2600**, Check Out **2602**, Payment **2604**, and Fulfillment **2606**.

Payment **2604**

After a total has been established, a payment method must be determined. A variety of mediums can handle the transfer of money. The methods, flow, technology, and potentially messaging, will vary by implementation. Issues concerning security, liability, and relationship to fulfillment need to be worked out.

Listed below are some considerations for determining the payment flow and mediums to be utilized.

Anonymity. If there is a need to allow the users to remain anonymous, an anonymous medium may need to be implemented. Implementations such as a silent bidding site may require strict standards and mediums for anonymity. In general, anonymity is not a concern for most implementations.

From the independent claims, a "system identifier" provides context for resolving a schema identifier that is part of an initiation or response message. Applicants do not find in any of the discussion of "Payment reference 2604", either in the passages set out above or in the following columns 63-64 any system identifier that provides context for resolving a schema identifier that is part of an initiation or response message. Claims 1 and 3 include less detail than the 2604 passages and, again, do not teach any system identifier that provides context for resolving a schema identifier that is part of an initiation or response message.

Therefore, Applicants respectfully submit that claims 2, 30, 48 and 65 are allowable over Bowman-Amuah.

Dependent **claims 19, 38, 56 and 73** require that the polymorphic response be represented in a form compatible with XML. The Examiner argues that a polymorphic response as an extension of XML "is inherent in object classes for programmers to mix and match different classes and create specialized objects using XML in a polymorphic response." MPEP 212 at 2100-54 to –55 (Rev. 2, May 2004) sets forth a strict rule for implying features into XML:

> "In relying upon the theory of inherency, the examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that the allegedly inherent characteristic necessarily flows from the teachings of the applied prior art." *Ex parte Levy*, 17 USPQ2d 1461, 1464 (Bd. Pat. App. & Inter. 1990) (emphasis in original).

Applicants do not understand the Examiner's reasoning on using XML in a polymorphic response. We do not see that the claimed system identifier, schema identifier, type and version, and additional data elements necessarily flow from using C++ as a programming language to generate XML messages. Inherency does not work here.

Therefore, Applicants respectfully submit that claims 19, 38, 56 and 73 are allowable over Bowman-Amuah.

Many dependent claims include limitations that are not addressed by any of the passages cited from Bowman-Amuah. Claims 3, 7, 9 and 11-28 depend from claim 1 and include limitations not addressed by Bowman-Amuah or in the office action. Claims 31, 35, 37, and 39-46 depend from claim 29 and include limitations not addressed by Bowman-Amuah or in the office action. Claims 49, 53, 55 and 57-63 depend from claim 47 and include limitations not addressed by Bowman-Amuah or in the office action. Finally, claims 66, 70, 72 and 74-83 depend from claim 65 and include limitations not addressed by Bowman-Amuah or in the office action. Therefore, respectfully submit that these many dependent claims are allowable over Bowman-Amuah.

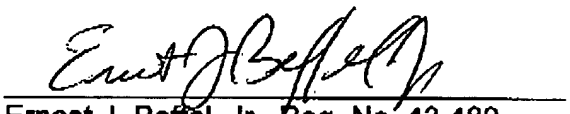
Application No.: 09/730,489 **Atty Docket: JGR/CMRC 1003-1**

CONCLUSION

Applicants respectfully submit that the pending claims are now in condition for allowance and thereby solicit acceptance of the claims, in light of these amendments.

If the Examiner disagrees with the analysis above, an interview is invited. The undersigned can ordinarily be reached at his office at (650) 712-0340 from 8:30 to 5:30 PST, M-F and can be reached at his cell phone (415) 902-6112 most other times.

Respectfully submitted,



Ernest J. Beffel, Jr., Reg. No. 43,489

Dated: 03 February 2005

Haynes Beffel & Wolfeld LLP
P.O. Box 366
Half Moon Bay, CA 94019
(650) 712-0340 (telephone)
(650) 712-0263 (facsimile)